EMGT 835 FIELD PROJECT:
Feasibility of SOA in a Legacy Environment

By

Naveed Anjum

Master of Science

The University of Kansas

Fall Semester, 2008

An EMGT Field Project report submitted to the Engineering Management Program and the Faculty of the Graduate School of The University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

Dr. Bob Zerwekh_____
Committee Chair Date

Annette Tetmeyer_____
Committee Member Date

Saad Siddiqi_____
Committee Member Date

Acknowledgments

I would like to first thank my committee chairman, Dr. Robert Zerwekh and committee members, Annette Tetmeyer and Saad Siddiqi, for the support and guidance they provided me for this field project. I would also like to thank the following teams who have assisted me greatly in completing this paper:

- Technical Architecture team that is heavily involved in implementing infrastructure components of the Service Oriented Architecture at YRCW
- Enterprise Architecture team, responsible for SOA implementation within YRCW

Executive Summary

Service Oriented Architecture (SOA) is currently viewed as the “silver bullet” solution for all of the integration issues that a business faces in this very competitive world. SOA technologies vendors would also like everyone to believe the “silver bullet” myth. While there is definitely some truth to the notion of SOA being able to solve many integration issues, SOA does not apply to all integration issues and it should not be considered as the only technology solution. The core argument is whether SOA is a technology or a methodology. Most experts will argue that it is a methodology with definite software architecture pattern, but many case studies have shown that it is a complete mindset change, a new ideology as to how software should be created to solve business problems.

This ideology of SOA has been around for decades in many forms and it is coming to fruition as the internet is creating the concept of a global information village. The global information village is giving people the opportunity to enhance communication and find solutions to all sorts of problems eliminating the need to recreate solutions. The idea of reusability is catching up very fast and that’s where SOA is claiming most of it benefits like speed to market and reduced total cost of ownership (TCO) for software development.

SOA is definitely not for everyone, especially for environments that are accustomed to legacy technologies and associated software development practices. There is a limited scope for SOA in legacy environments and not all problems can be solved with SOA. For instance if a company already has a monolithic system that is performance oriented and is meeting the needs of the business, then future enhancements will never dictate the need for SOA as the initial investment into SOA is very high. From a performance perspective, a monolithic system will always be faster since network latency due to dispersed and segregated multiple computing engines integration is the nature of SOA. A good application of SOA in the legacy environment would be e-commerce where monolithic system capabilities can be abstracted out and made web enabled. In this approach one does not have to rewrite its monolithic system but only use SOA to define a new interface to interact with the monolithic system.

Table of Contents

Table of Figures

Acronyms and Definitions

API	Application programming interface
Batch	Opposite of real time processing of transaction i.e. not real time
EAI	Enterprise Application Architecture
Enterprise Architecture	According to Wikipedia “is the organizing logic for business processes and IT infrastructure reflecting the integration and standardization requirements of the firm’s operating model.”
Integration Layer	A software layer that abstracts backend functionality into reusable interfaces
Interoperable	Ability to exchange information with other systems easily i.e. a computer module can be used on different platforms and computer languages
KPI	Key performance indicator
Loose Coupling	A system is considered loosely coupled when it does not require minimum or no change when it interfaces are changed
Middleware	Software component that serves as an abstract layer or intermediary between systems
Monolithic	Silo system where are the functionality is done within a single computing engine
Order to Cash	Business process model which start from order entry to cash receipt
Rationalize	To choose or select a system among available redundant systems
SDLC	Software development life cycle
SLA	Service level Agreement
SOA	Service Oriented Architecture
TCO	Total Cost of Ownership
Transaction Management	It is the concept where a transaction is ensured to complete fully so that the integrity of the data is maintained

Introduction

In a world where we cannot achieve different objectives without interacting with each other, the computing world is also dependent on the same principle. With the advent of the computer, there have been specialized components created that are residing on separate computing engines all over the world. In earlier days, computing was silo based with mainframes running all the applications that departments or businesses needed for a particular business function and most of the interactions were done manually using input devices. As the computing industry has matured, the need to access data and related functions and process them in real-time has become a necessity to be competitive in the marketplace. This gave rise to the term "distributed system" and "systems integration" which in essence means that multiple physical computing engines are located in different places and the actual system functions only by integrating these disperse systems into a logical single unit. In other words, distributed systems were created as a result of recognizing the fact that in order to automate business functions data and related functions' aggregation is needed which are normally not located in a single physical system. In business terms there is a business process that is trying to achieve a business objective by relying on multiple other business processes by integrating with them.

There are also many other terms in the software industry that are used to describe integration [5], one of these terms that is very common is Enterprise Application Integration (EAI) [5]. The idea as to what EAI means can be very easily depicted in a diagram. Figure 1 describes on a high level what EAI means. To fully understand the value of EAI you have to compare Figure 1 with Figure 2. The diagram comparison clearly shows that in the absence of EAI, each system has to communicate with every other system if it needs any data from that system. With EAI, a system will only have to talk to the middleware (EAI) to get any data

from a system that is already integrated with the middleware. This pattern of EAI is of great benefit to any computing environment where data and business functions are dispersed into multiple physical or logical locations.

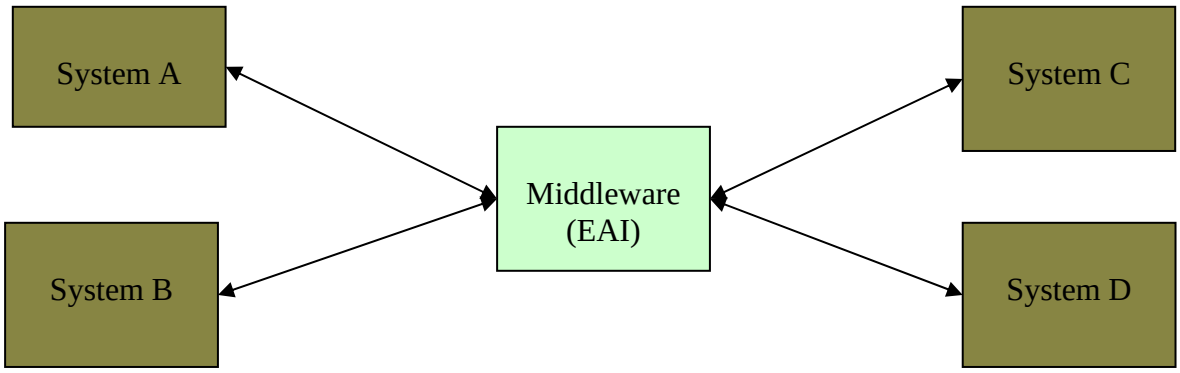

Figure 1 – EAI Pattern

On a high level some of the advantages and disadvantages of EAI are:

- Advantages
    - Reusability i.e. develop once and used by many
    - Support both real time and batch information access among systems
    - Raise organizational efficiency by streamlining business processes
    - Information integrity across multiple systems through transaction management
    - Ease of development and maintenance
- Disadvantages
    - Possibly high development costs, especially for small and mid-sized businesses where there are few dispersed systems
    - EAI implementations are very time consuming, and need a lot of resources
    - Require a fair amount of up front design, which many managers are not able to envision or not willing to invest in

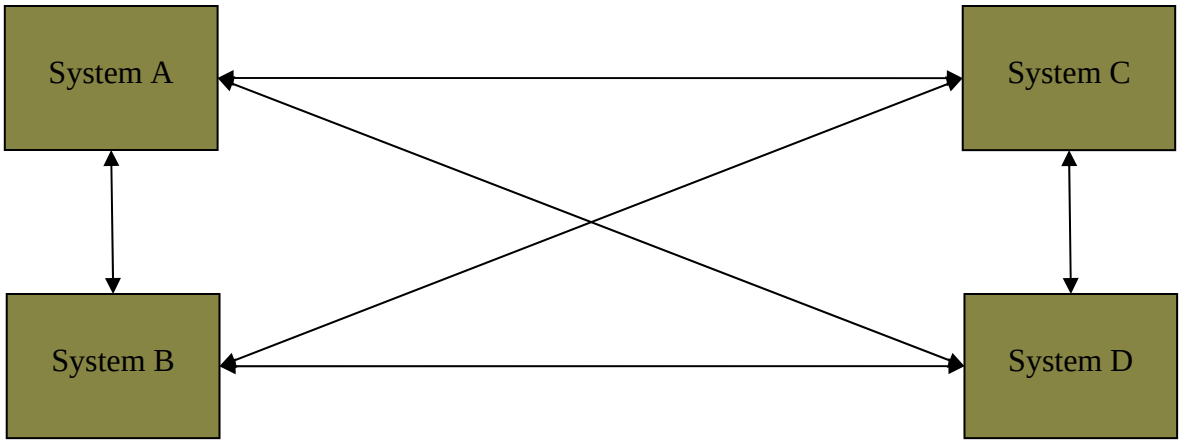

Figure 2 – Point to Point Integration

This introduction to distributed systems and EAI is necessary since the goal of this paper is to show the feasibility of implementing Service Oriented Architecture (SOA) in a legacy environment. Legacy environments would be any corporation's IT environment that is leveraging very old technologies like COBOL, JCL, Adabase, etc. along with below average application development practices. While these technologies and application development practices have definitely improved as years have passed, their initial purpose was for monolithic or silo systems typically implemented in an environment with minimum or no integration and following the old waterfall methodology for implementing IT projects. In most cases, integration is batch and not real-time. From a timeline perspective Figure 3 documents the evolution of the integration architecture paradigms. Defining each paradigm is out of the scope for this paper but it is important to understand that the IT industry has recognized the need for integration and has been working on it for a long time. SOA is the latest approach in this wave of determining the best way to integrate systems.

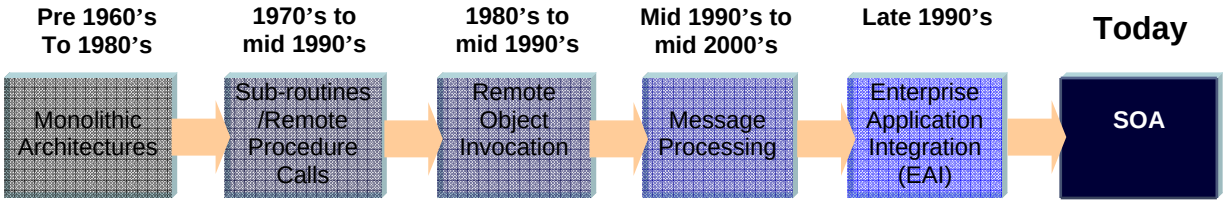

Figure 3 – Integration Evolution

An additional goal for this paper is to determine whether SOA is the "real thing" (as IT vendors claim in advertising) or if there is more to SOA than the normal IT professional understands. Sun Microsystems SOA Architect [8] uses the diagram below to depict the change SOA can bring into an environment.

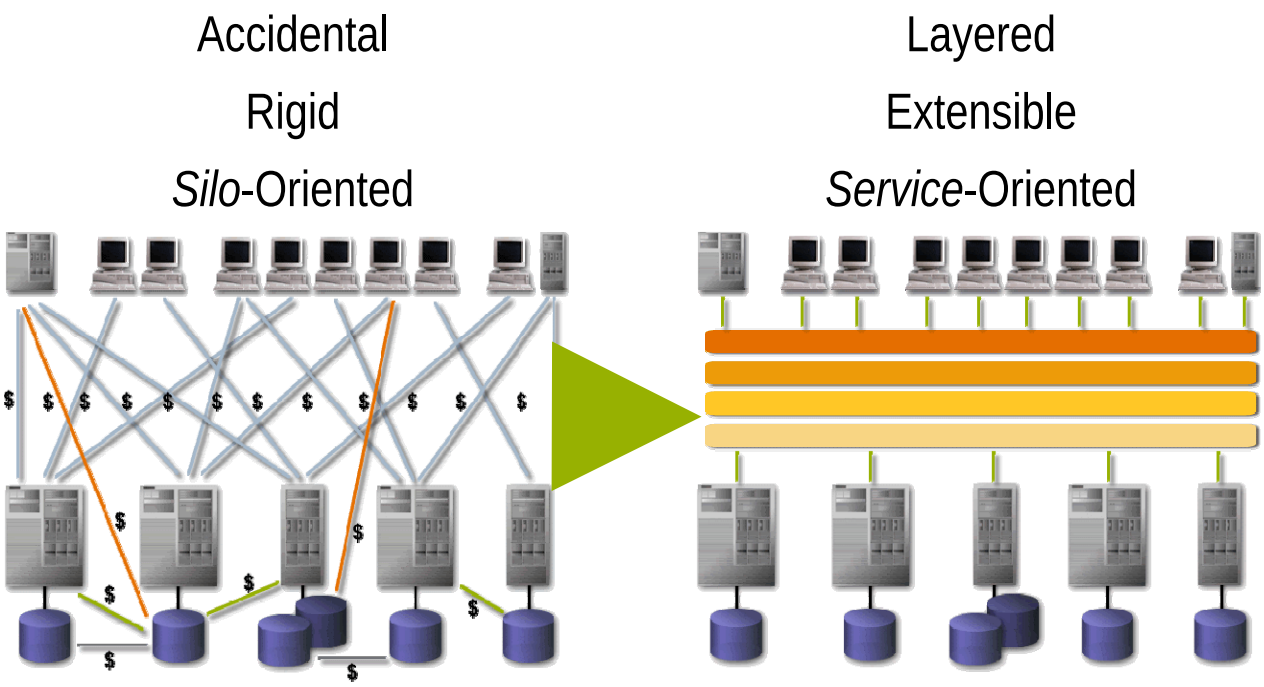

Figure 4 – Silo Oriented vs. Service Oriented Architecture

Keeping this diagram in mind, the question arises as to whether SOA is the only solution or if there are other alternatives to solve the silo-oriented architecture issues. Therefore, the topic "Feasibility of SOA in a Legacy Environment" is critical to the computing industry. IT shops all over the world try to follow the technology adoption wave

every time a new term is created with huge capital backing from industry giants such as Microsoft, IBM and others. IT professionals all over the world are trying to understand whether SOA is just a buzz word or if SOA is really a true solution that can benefit businesses.

To begin, SOA must be defined. According to one of the leaders in the SOA space, IBM defines SOA as [1]:

> *"Service Oriented Architecture (SOA) can mean different things to different people depending on the person's role and context (business, architecture, implementation, and operational). From a business perspective, SOA defines a set of business services composed to capture the business design that the enterprise wants to expose internally, as well as its customers and partners. From an architectural perspective, SOA is an architectural style that supports service orientation. At an implementation level, SOA is fulfilled using a standards-based infrastructure, programming model, and technologies, such as Web services. From an operational perspective, SOA includes a set of agreements between service consumers and providers that specify the quality of service, as well as report on the key business and IT metrics".*

BEA, another major vendor of SOA related technologies describes SOA as [2]:

> *"Enterprises that effectively align technology with business goals achieve competitive advantage. BEA is helping enterprises to achieve faster time to value by driving compatibility across the application infrastructure through open standards and service-oriented architecture (SOA). Service-Oriented Architecture is an IT strategy that organizes the discrete functions contained in enterprise applications into interoperable, standards-based services that can*

be combined and reused quickly to meet business needs. By organizing enterprise IT around services instead of around applications, SOA provides key benefits:

- *Improves productivity, agility and speed for both Business and IT*
- *Allows IT to deliver services faster and align closer with business*
- *Allows the business to respond quicker and deliver optimal user experience*
- *Masks the underlying technical complexity of the IT environment"*

Microsoft [3] has gone with a very generic definition:

"SOA is a standards-based design approach to creating an integrated IT infrastructure capable of rapidly responding to changing business needs. SOA provides the principles and guidance to transform a company's existing array of heterogeneous, distributed, complex and inflexible IT resources into integrated, simplified and highly flexible resources that can be changed and composed to more directly support business goals".

Wikipedia [4]**,** one of the internet most powerful collaboration and knowledge repository documents has a different definition:

"*Service-oriented architecture (SOA) is a method for systems development and integration where functionality is grouped around business processes and packaged as interoperable services. SOA also describes IT infrastructure which allows different applications to exchange data with one another as they participate in business processes. The aim is a loose coupling of services with operating systems, programming languages and other technologies which underlie applications. SOA separates functions into distinct units, or services,*

which are made accessible over a network in order that they can be combined and reused in the production of business applications. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services. SOA concepts are often seen as built upon, and evolving from older concepts of distributed computing and modular programming".

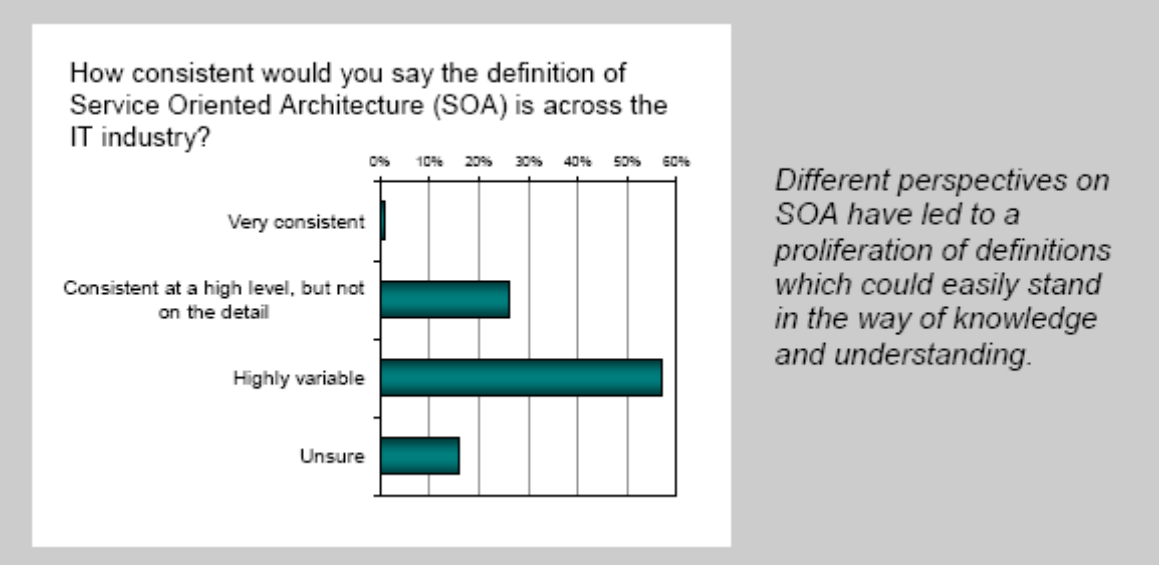

Figure 5 – Freeform Dynamic (IT research firm) - SOA definition consistency survey

There are other definitions with some variations for SOA which are causing some issues in the IT industry as shown in Figure 5 [17]. In the author's opinion, the definition above from Wikipedia provides a holistic summary of what the mainstream information technology professionals are striving for under the SOA umbrella. From a diagram perspective, IBM's SOA stack [10] provides a good high level definition as shown in Figure 6. The stack clearly distinguishes the different layers in the SOA life cycle both horizontally and vertically. Horizontally the service creation aspects are visible and vertically how SOA should be managed is defined

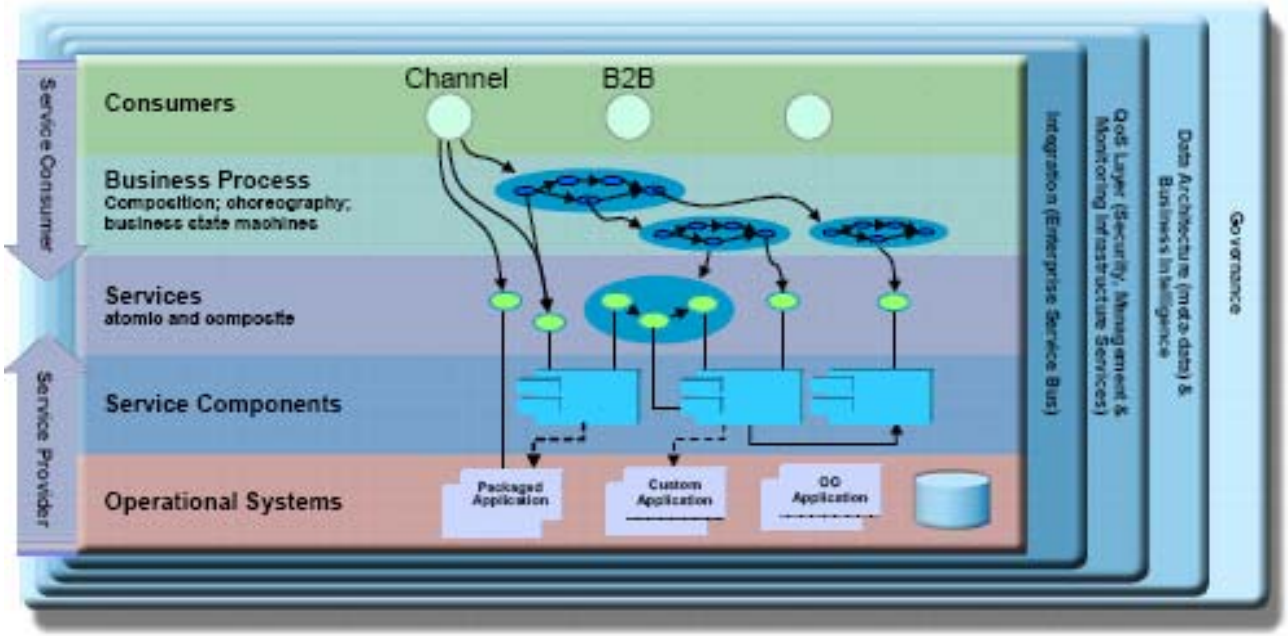

Figure 6 – IBM SOA Stack

On a high level, the actual SOA specification comprises of the pattern shown in Figure 7. The key components of which are:

Service Provider: the entity that accepts and executes requests from consuming applications.

Service Consumer: the entity that calls the provider for service functionality.

Service Registry: directory of the available services with attached contracts.

Service Contract: the way a service is available to a consumer for interaction.

The flow of the pattern is basically that a Service Provider publishes a service with the Service Contract in the Service Registry. The Service Consumer locates the service by finding it in the Service Registry, binds it with the Service Contract specification and then executes the service function.

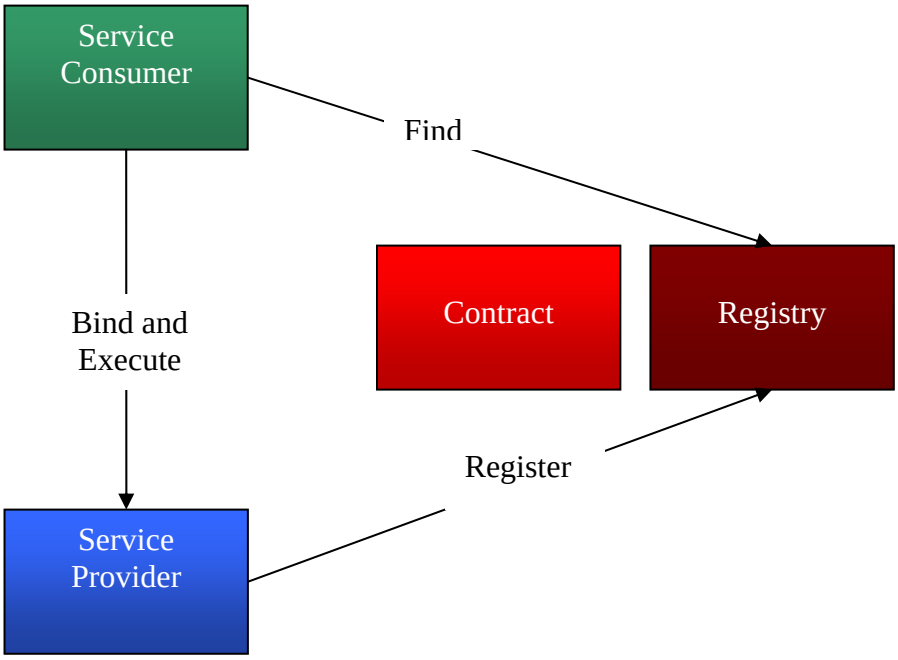

Figure 7 – SOA Pattern

On a characteristics level [11], SOA defines the Service as:

Services are discoverable and dynamically bound: meaning that a consumer application can figure out by querying a registry what services are available and then binds to the appropriate service through the interface definition that is available through the registry.

Services are self-contained and modular: meaning that the service itself is a unit of work that is multiple services can easily be combined to create an application.

Services stress interoperability: the idea that it could be available to be used on different platforms and computer languages.

Services are loosely coupled: meaning few well-known dependencies so that it can easily be enhanced when needed.

Services have a network-addressable interface: this point to the idea of being easily accessible from anywhere.

Services have coarse-grained interfaces: this is relative to the requirements of a project.

Services are location-transparent: this is a key concept of SOA where the consumer does not know where the service is located until they use the registry to get to it.

Services are composable: this point towards the modularity feature of the service that is aggregation of services to create application or composite service.

Service-oriented architecture supports self-healing: the ability to recover from errors without human intervention during execution.

The SOA wave is making a lot of noise because the benefits that are being advertised hit the essential goals of any competitive business today. These benefits are simple and powerful. They are

- Speed to market
- Reduce TCO

SOA is betting on speed to market based on the assumption that if the core capabilities that are needed by the business are SOA enabled, then creating business application becomes really easy in that multiple services can be aggregated to form a business application. TCO is reduced in the same process because SOA is hoping that less time and resources are needed to create the business application if services are already available.

Coming back to the goal of the paper, as experienced IT professionals who have seen the different waves of integration technologies and have participated in several integration projects as technical experts, are they truly doing their organizations a favor by just looking at the technical aspects of SOA? Is SOA just a technology or does it require other skill sets to

truly evaluate its benefits? IT organizations where enterprise architecture capabilities are considered critical and the enterprise architect role is defined and implemented are places where SOA is being challenged not only from a technical but also from a non-technical perspective. As mentioned before, the concept of SOA has lived in the past in different forms, enterprise architect are trying to answer whether the new SOA technologies and their associated implementation patterns are worth pursuing. The biggest question that has challenged new technologies like SOA is the concept of total cost of ownership (TCO). The IT industry has become accustomed to updating itself with new technologies in a rapid fashion and new technologies are continually emerging. When the overall economy was doing well in the past, there was enough capital for companies to invest in new technologies as they believed that new technologies would provide them with the competitive advantage in the marketplace. The slowing down of the economy has brought new mindsets which are asking for tangible benefits. SOA faces this new reality of being asked to prove itself.

Literature Review

In this section, three case studies and statistical data on SOA from different IT research firms are documented. These case studies are very important in understanding how SOA is viewed by the IT industry. A key point to note in these case studies is that company's legacy components are being targeted for transformation. On the other hand the statistical data shows how legacy environments are responding to SOA.

Industry Case Study # 1: EBS Building Society [6]

"EBS Building Society, Ireland's largest mutual building society, arrived at an SOA model of delivery by accident. "We have been doing integration and service work for years. We did not call it SOA until we discovered it written up recently and realized we were doing the same thing," says David Yeates, senior manager, IT architecture, at EBS. EBS runs its mortgage origination systems as a portfolio of services. After all, says Yeates, "A mortgage is actually a portfolio of systems blended into a single product." Over the years EBS had built general and life assurance systems, and had duplicated code and functionality as it went along. "We realized we could break down this monolithic application and give some of it to partners. It is no different from what manufacturers have been doing for years." The building society stumbled across a way of making this happen when it did an internal survey of reuse. "Our big moment came five or six years ago when we moved from simple, internal integration to rich integration. We had a proliferation of hardware and application servers - it was spaghetti." However, EBS had difficulties in achieving reuse, even though it had been dabbling in object oriented programming. "Investigation revealed

that the area where there was the most reuse was an ancient Cics application, which we had exposed to some terminals," says Yeates. The building society also discovered that keeping an audit trail was a good model for delivering services, according to Yeates. As a result of these discoveries, in 2001 EBS rolled out its Enterprise Application Integration Project, moving from a tactical to a strategic delivery. This encompassed building application and peer-to peer communication into its applications at the outset. At some point during this evolution, Yeates and his team discovered they could talk to business users using this communication technology. A non-IT literate business user was able to track and discuss the development of a service by observing the flow charts and diagrams on the wall in the IT department. "We were sharing the same vocabulary. But if we had talked about SOA, we would have been laughed out of the room," says Yeates. During the experimental and somewhat accidental business of implementing SOA, data semantics was the biggest stumbling block, according to Yeates. "The meaning of 'salary', 'name' or 'income' change according to who you are talking to. Nor do you have one system to deal with, but multiple ones, and that has huge implications for the IT department." There are a number of other important lessons that Yeates can pass on, too. He is keen to point out that SOA is not the same as web services. "We were doing SOA long before web services came along, using advanced program-to-program communication, remote procedure calls, remote method invocation and internet inter-orb protocol." Nor is SOA good for everything. "We got carried away at one point. We use expert systems to generate answers to online questions, but realized exposing these as a service was not appropriate." The biggest ongoing challenge for EBS is security. "If an external partner is providing part of your web service in real time they need to be working to a

framework such as ITIL," says Yeates. "Exposing your application in a business-to-business web service is a security headache that we are still finding our way around."”

Industry Case Study # 2: Verizon goes back to the workbench[7]

“To overcome its SOA roadblocks, Verizon had to build an entire SOA operational infrastructure virtually from scratch -- and it has the patents to prove it.” As a technology, Web services are great, but today's standards don't have nearly enough operational infrastructure around them," says Shadman Zafar, Verizon's senior vice president of architecture and e-services. "You can end up with a plethora of Web services but no awareness of which of them are where and provide what function -- and most important -- which have the right kind of capacity and SLA to be usable by what and whom. The result is that SOA risks simply becoming a toy for the developer." As have many other SOA implementations, Verizon's started after a merger -- in this case, the GTE/Bell Atlantic merger that created Verizon. "I was tasked to integrate the two IT departments to achieve strong synergy targets," Zafar says. "During our initial research, we found that many of our core business functions were implemented anywhere from five to 30 times across different applications." Verizon set out to use SOA to reduce support costs by consolidating these core functions from 20 or 30 implementations to one or three, in each case. The first step was to spend months identifying roughly 500 key business functions that were used over and over again for many applications. "We didn't go through 10 million lines of code. We picked out 500 business functions and targeted them," Zafar says, citing examples such as the credit check service, the telephone number engine that provides new customers with telephone numbers, and the address validation service. "An ordering system might go

through four or five of these key components." Getting these 500 Web services built required deploying SOA not just as a methodology, but as an IT governance principle. "We implemented a process whereby all development projects would have to go through a strategic architectural assessment session to get approval before they were started. If any of the functions were one of those 500 core business functions we had identified, or we found another suitable business object, the team would have to implement it as a Web service.

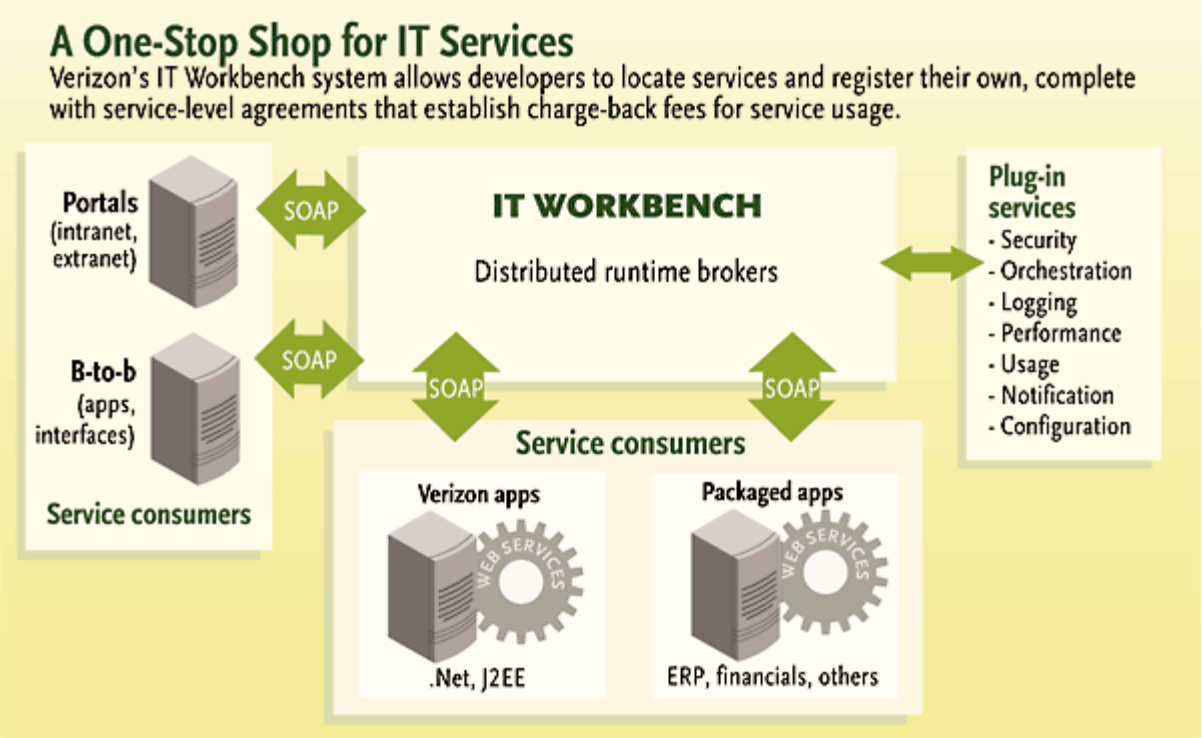

Figure 8 – Verizon SOA implementation

Once the project was completed, it would go through a tactical architectural assessment session that would verify the Web services were built before giving clearance to go into production," Zafar says. But as developers began building Web services, the operational roadblocks became clear. "There was a lot of resistance to using the services because of the lack of standards around four essential operational pieces: SLAs, accounting, billing, and capacity management. People weren't willing to

entrust their mission-critical applications to each other," Zafar says. Developers were also very reluctant to expose the code they had worked so hard to produce. In many cases, one developer wouldn't even know what infrastructure the other Web services were using. To address these issues, Verizon built IT Workbench, an infrastructure layer that houses the Web services directory but also adds management capabilities. For example, if one development team built an address-validation service that it wanted to expose to other applications, it could download a lightweight agent onto its server that would register the Web service and its capacity and define an SLA inside IT Workbench. After services have been registered, developers can go to the central directory to find ones with their specific requirements. For example, they can search for a service with the capacity to support 100,000 transactions per day with less than a two-second response time and a commitment to be available 99.8 percent of the time from 8 a.m. to 12 p.m. Accounting and billing are also in place, so the service provider can charge for usage -- another incentive to use SOA. "It started with onesies and twosies," Zafar says. "But when it hit 10,000 transactions per month, [SOA] suddenly took off with exponential growth. Now it's used in almost 10 million transactions per day." What's more, any reluctance to expose code has virtually disappeared. According to Zafar, developers are now competing to get others to use their services, as a way of gaining recognition within the company. The most-used services are listed on the IT Workbench portal with the author's name and photo. "Before, people would say, 'This is my code, use your own,' "Zafar says. "Now they're reaching out to each other over the weekends, saying, 'Why don't you use this service I built,' so they can be more popular on the IT Workbench portal. In fact," he adds, "developers are trying to push applications as Web services that are not suitable

because they have few or no logical consumers. It's an interesting social phenomenon that I never anticipated."

Industry Case Study # 3: Austin Energy - SOA project gets mother of all stress tests [20]

"Improving call center productivity and consequently customer satisfaction were the two goals the CIO had when his team of 21 developers began planning their initial SOA implementation last fall. While the goal is to eventually replace more than 70 legacy systems with SOA applications, Carvallo followed the advice to begin incrementally with a small project. The customer service implementation is a composite application with five Web services designed to handle actions such as verifying the customer location and generating a work order to repair the outage. Moving into the SOA world required not only a new way of thinking about application development, but a complete updating of the programming skills for the development team. Carvallo had determined that the new SOA applications would be written in Java using IBM's WebSphere and Rational tools, but the developers at Austin Energy were experienced in working with Cold Fusion and Visual Basic. To transform his 21 developers into Java coders, Carvallo sent them to an IBM facility in Austin for 60-days of Java training. With no previous experience designing SOA applications, the CIO brought in architects from IBM Services to do the architecture and oversee the development. However, the actual coding was done by his 21 newly-minted Java programmers. The combination of outside consultants and in-house coders took only six months from concept to implementation and produced the application that rode out the storm on May 4. This was particularly gratifying to Carvallo because it disproved the naysayers who told him it couldn't be done. "A lot of the people at the beginning

thought this could never happen," he recalled. "They said it will take forever. Obviously we proved that it can be done. It's not impossible to move to this world, do it right and deploy something that scales better, is more modular and achieves all the requirements on time and on budget." One of the keys to success, Carvallo said was that 30 percent of the project was focused on planning and the architecture. The 60-days of Java training leading to certification for the development team also paid off, he said. Now that team is moving on to the next SOA application. "The end goal is to have an integrated enterprise with SOA in the middle of it," Carvallo said. "We have been mapping out since last summer the key processes – 70-plus processes – that we want to bring into the SOA (environment) and help rationalize the infrastructure behind it. We will have the next five processes tackled in the next five months and go live with a set of new Web services and deployments and extensions of the SOA to tie to things like the financial system and the asset management system.""

These three case studies are very unique because they identify three main categories of how SOA is being implemented in different organizations

- Implementation of SOA without knowing it
- Building custom SOA implementation
- SOA Vendor solution

The key points out of these cases studies were

- SOA is all about modularization
- SOA requires a good capital investment
- SOA does not necessarily require SOA specific technology tools

- Reusability is key to SOA
- SOA needs executive sponsorship and leadership
- Vendor SOA definition is not what SOA is being defined as in the industry

SOA Statistical Data

There are many IT research firm that specialize in gathering data on how new trends in the IT industry are being adopted. Aberdeen Group is one such IT research firm which regularly publishes statistics on how well a technology is being implemented and what are its implications in the industry. The data below uses the following acronyms that will be needed to understand the statistics.

- LAG: This means “Laggard” which Aberdeen describes as IT organizations whose application development practices are significantly behind the average of the industry. This definition basically applies to many IT department that are dependent on legacy environment. These could be IT organizations that have good application development practices but still leverage legacy technologies. These instances are very small in number and are not the norm in the industry.
- AVG: This means “Average” which Aberdeen describes as IT organization whose application development practices are average or norm. These organizations have basically invested in new technologies and are adjusting their application development practices accordingly to be competitive in the marketplace.
- BIC: This means “Best in Class” which Aberdeen describes as IT organizations that are the best currently in the industry. These organizations have heavily invested in lowering their IT integration complexity,

standardizing and simplifying development structures and proactively measuring key performance indicators (KPI) to manage their projects.

In Figure 9 [19] it is evident that companies with legacy environments are moving towards SOA. This movement compared to other companies where newer technologies and software development practices have matured in time is much limited. Also from a cost perspective Figure 10 [19] shows that companies with legacy environments are not seeing any cost reductions in development. The benefits to the companies with legacy environment are minimum in the area of maintenance and lifecycle costs and there are considerable number of companies whose cost have either remained same or gotten worsen with SOA implementation.

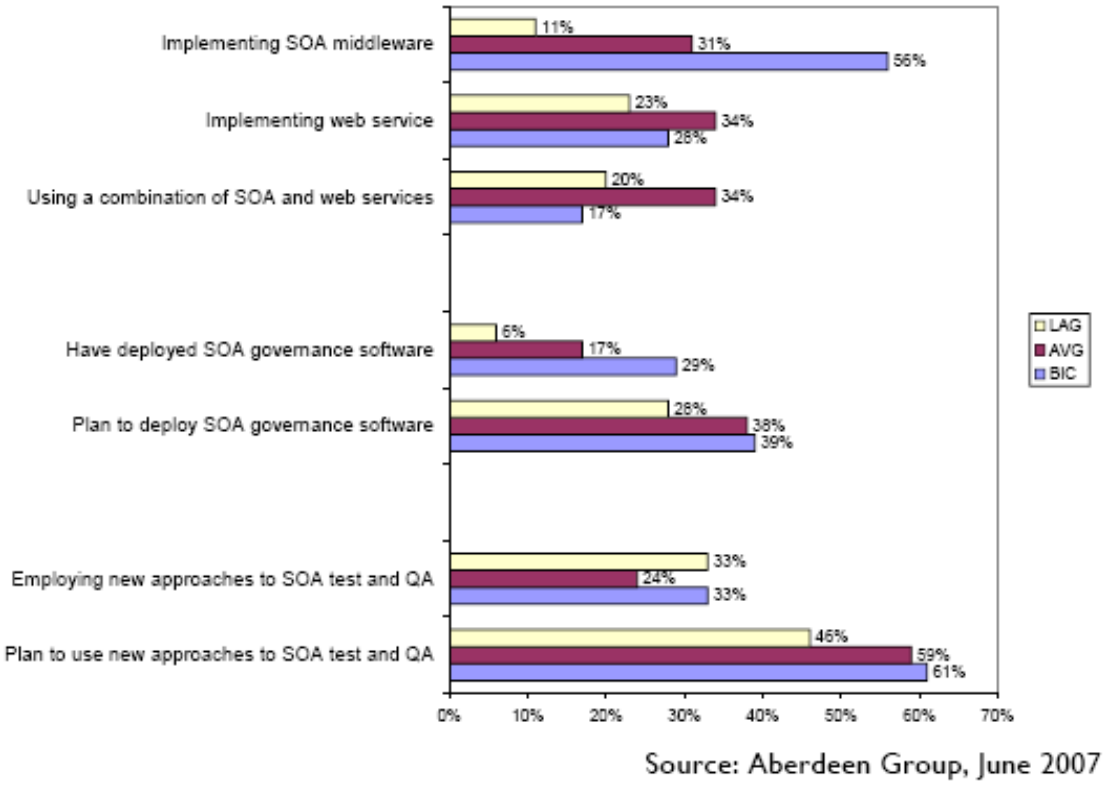

Figure 9 – Aberdeen Survey result on SOA implementation

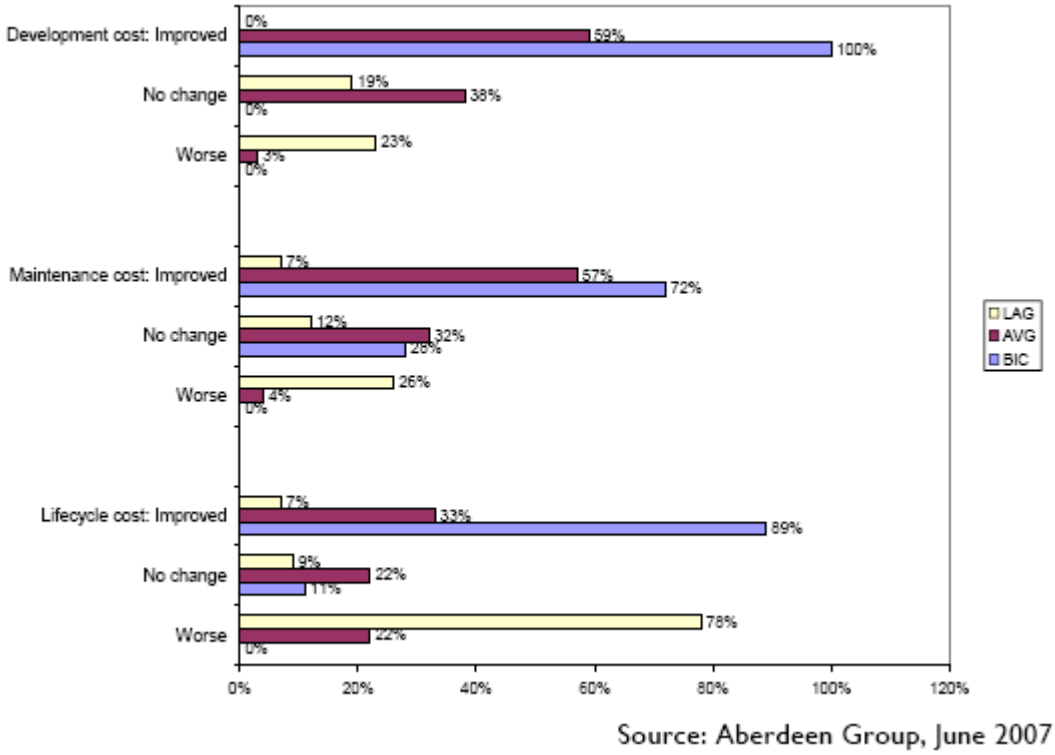

Figure 10 – Aberdeen Survey on SOA costs

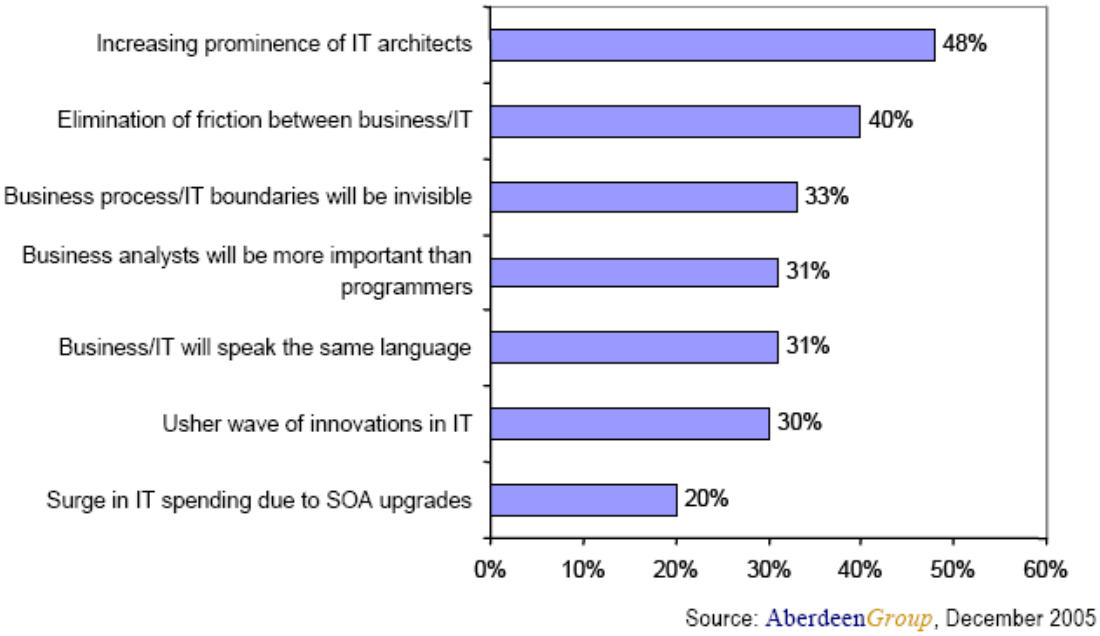

Figure 11 – SOA impact on business and IT

SOA is also doing a great service in removing the old friction between IT and business, Figure 11 [18] documents that new relationship between IT and business. From a cost perspective it also shows that IT spending is definitely increasing due to SOA.

Projects Review

In this section, analysis of five case studies are documented. They are directly from SOA projects where the project team members were available to provide information on the scope, requirements and solution of the projects. Due to confidential nature of the information in these case studies, company names and related data that is considered proprietary are not published.

Project Case Study # 1: Customer Trouble Ticket System

Background: In a very competitive marketplace where businesses are trying to provide additional value added services to sell their product to other businesses, this case study deals with an environment where to promote a product, a company is providing free access to its internal trouble ticketing system which will enable any business customer to directly interact with the company to open trouble tickets. This access will remove the interaction a business customer will normally have by calling the help desk or customer care center for that product plus it will provide proactive status and updates to the customer as to how the issues are being managed. In other words, a business customer can open a ticket with the company much faster and the company can reduce cost by eliminating the need for an extensive call center environment for that product support. They will still need some level of customer care but the direct access to trouble ticketing system will reduce the headcount needed.

Scope: This value added service is only targeted to big businesses that are high revenue customers for the company and have the necessary infrastructure in place to integrate with the trouble ticketing system. The main requirement for the project was that the capability is created in such a fashion that it is highly reusable. In other words, any customer that is chosen

can leverage it without any additional customization and the interface is published in a fashion that a business customer can easily integrate it into the IT environment.

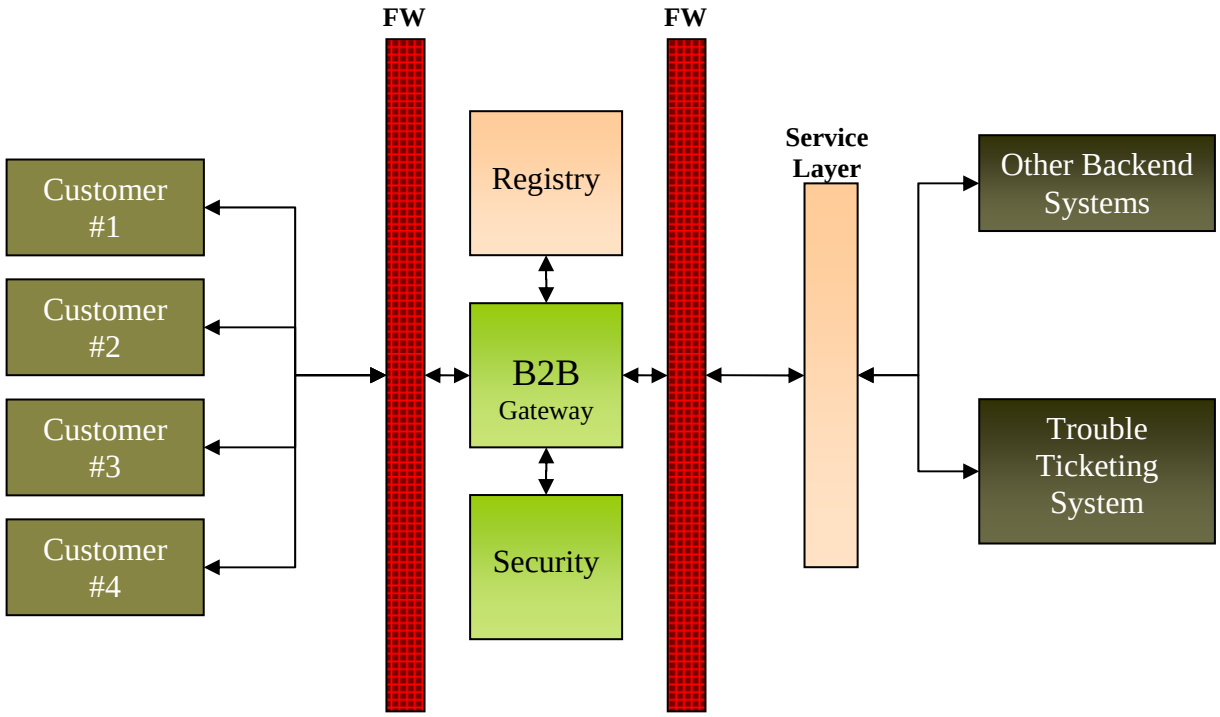

Figure 12 – Trouble Ticketing System SOA enablement

Solution: The Company chose to adopt the service oriented architecture pattern to solve this issue. The main reasons behind the decision were:

1. Service will be a highly reusable component since it will be created from scratch with the new technology based on industry standards
2. The existing environment within the company did not have any other technology that could meet the reusability, ease of access and security requirements
3. The actual ticketing system already existed for internal use but it needed an abstraction layer that could expose the essential capabilities to the outside world with necessary security

4. A public registry for services was need which the SOA pattern provides
5. There were minimum budget constraints since that the company's product was very profitable i.e., good business case

From an IT development perspective, only the integration team was impacted. This team worked with the business to lock the scope and implemented the project in record time even though the vendor recommended governance model for SOA was not in place with the software development lifecycle of the company. This project did not use the normal software development lifecycle process but instead the IT side let the business take the lead on managing the project leaving IT to provide technical resources only. The project team indicated that if the project had gone through the normal software development lifecycle then they would not have been able to implement this in a timely manner and the cost of the project would have been considerably higher.

Conclusion: The following was learned from this project:

1. There was a very good business case and there were no budget constraints.
2. The project team was able to buy the best tools available.
3. Backend capability already existed; therefore the service was just an abstraction layer.
4. SOA requires a good governance model as most of the industry vendors are recommending but in this case the project was lucky since only one IT team was impacted and they did not use the normal software development lifecycle process.

Project Case Study # 2: Order Management System

Background: This case study dealt with a merger environment where two companies with different IT philosophies were merged. Company #1 is completely dependent on a billing vendor who actually manages the majority of the "Order to Cash" processes for the company. On the other hand, company #2 has a hybrid solution where the majority of the customer facing IT applications are completely independent of the billing system. Figure 13 and Figure 14 show the high level architecture of the two companies.

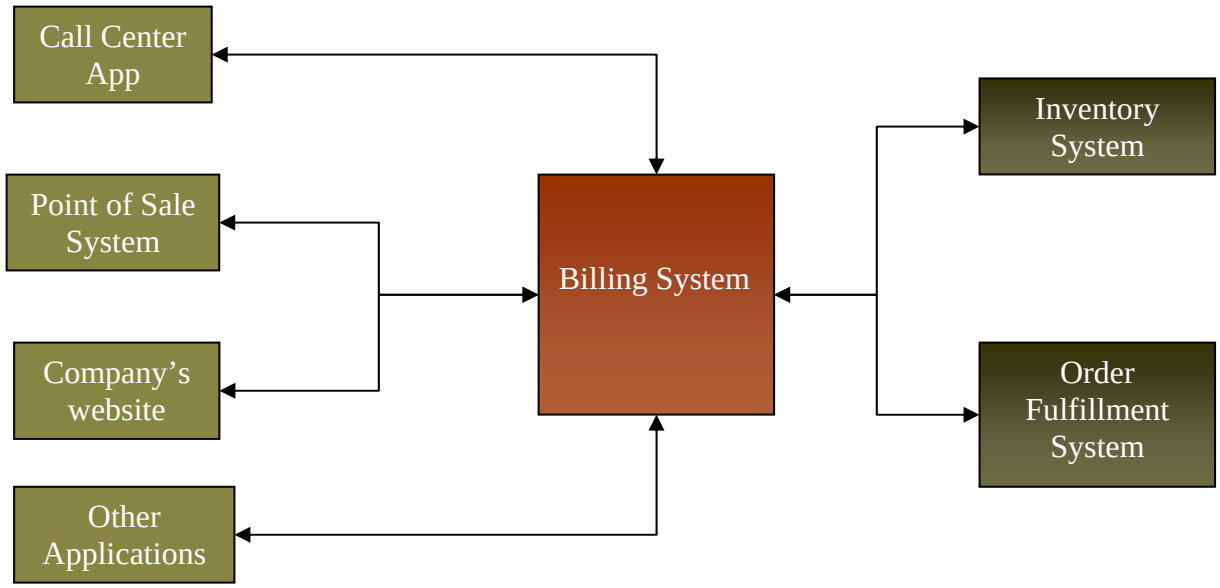

Figure 13 – Company #1 - Vendor proprietary order management system

Figure 13 shows a point-to-point integration of the different applications into the main backend system. This pattern basically locks the architecture to a particular vendor and it becomes difficult to add other vendor solutions or even in-house components due to the proprietary, non-standard based architecture. In this kind of environment it is very critical that the contract negotiated with the vendor protects the company holistically. There has to be a stringent service level agreement (SLA) defined to ensure the performance and availability of the solution as well as ensure timely vendor responsiveness to issues.

Company #2 solution is a little more modular in nature where an integration layer was created to ensure that the architecture in place was flexible and performance oriented. Figure 14 clearly shows the pattern that the backend systems are completely independent from the front-end systems. In other words, one can easily add or remove systems from the environment since the integration layer abstracts the actual system interfaces and provides a standard interface for all systems to interact with each other.

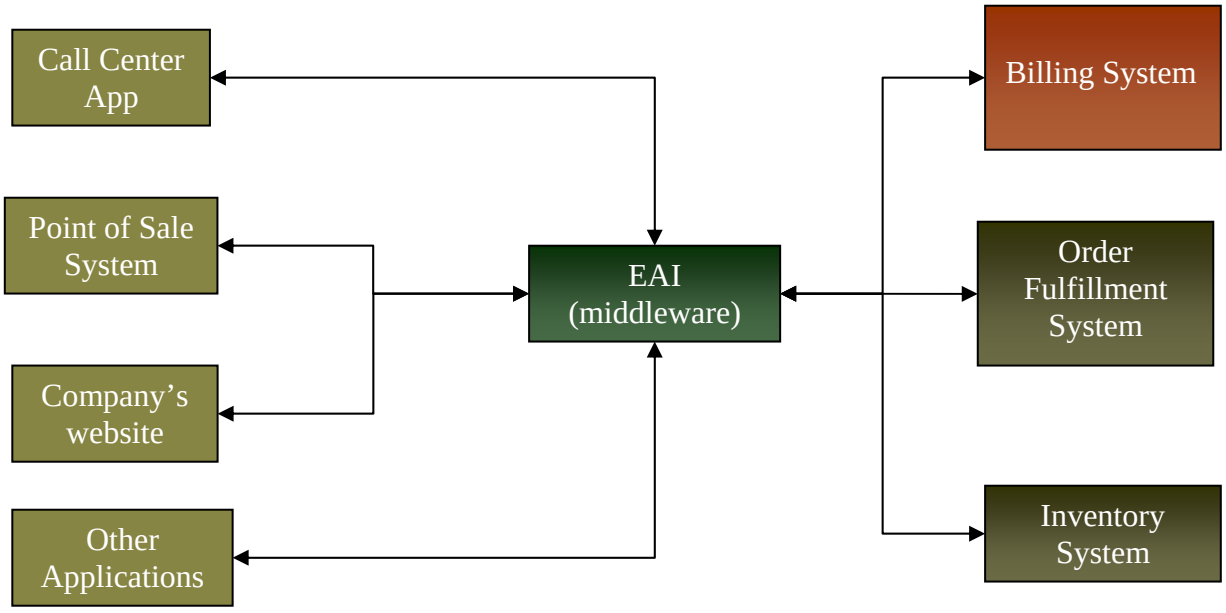

Figure 14 – Company #2 – Abstract integration layer

Scope: The scope of the work that was needed for this project was to first rationalize the IT application portfolio and then figure out how to integrate them. The new merged company chose the billing system from company #1 and other systems from company #2. The rationalization of the IT application portfolio created a huge issue as to how to integrate the chosen applications since the billing system will be the integration point.

Solution: From a theoretical perspective, the company #2 architecture was more modular and this aligned better with the SOA model. From a long term perspective, this architecture could

have served the merged company better as it would be easier to add more components in the environment which is critical in the current global competitive environment. The merged company chose to go for more a monolithic architecture like company #1. They actually had valid reason for this and they were:

1. Billing vendor was promising an overall cheaper solution.
    a. This was in alignment with the merged company's vision of outsourcing since the vendor was planning to use off-source resource for this implementation and was able to quote a cheaper implementation initially.
2. From a timeline perspective it made sense to adopt company #1 architecture paradigm since it would require less integration if minimum company #2 systems were chosen.
    a. The merged company reversed some of its decisions to adopt company #2 system in order to meet the timeline.
3. Both companies were accustomed to the waterfall methodology for software development lifecycle; therefore modularization of components would be very costly since most of the chosen systems were not modular.

Conclusion: After this project was implemented, the merged company encountered many issues including:

1. Major data migration issues; moving data from a heterogeneous environment to a monolithic environment is very challenging since just identifying what data needs to be moved is much harder.
2. All the legacy customer capabilities were compromised due to the limitation of the vendor solution.

3. The load expectation on the merged system was not managed correctly since load and performance testing timelines were compromised.
4. The off-sourcing of critical business supporting IT functions required very prompt response which the new environment lacked.
5. The contract with the vendor was missing key contingency elements to protect the merged company (like stringent SLAs).
6. The new merged company was having a major issue in introducing new products since the software architecture was not flexible and was locked with one single vendor.

Project Case Study # 3: Customer Care System

Background: This case study deals with a startup company which did not have any IT infrastructure in place to interact with its customer. The company was a reseller of services and had only 6 months to put in place an "Order to Cash" system in place to interact with its customers.

Scope: As mentioned the main requirement was to put in necessary IT infrastructure to enable the "Order to Cash" process.

Figure 15 – Monolithic customer care system

Solution: There were multiple options available, but the company chose to implement an outsourced CRM solution which provided the necessary graphical interface for the call center to take orders from customers, collect payments, provide status on orders, open trouble tickets

and enable services. These processes can individually be considered services but they were tightly coupled with each other since the CRM package did all this functionality without exposing any interfaces to interact with individual services independently.

Conclusion: For a small company with a limited budget the solution selected was the most feasible since the situation demanded a tactical solution to launch the company as soon as possible.

Project Case Study # 4: Intranet Portal Implementation

Background: This case study also deals with a merger IT environment. Here instead of two companies coming together, there were four companies that were trying to consolidate IT infrastructures. The parent company had been buying other companies without integrating them, i.e., after the purchase these individual companies were allowed to operate independently thus the overall corporate environment had redundant systems. As the global economy was becoming very competitive, maintaining redundant systems did not make any sense from a total cost of ownership (TCO) perspective. Therefore as an initial step to start consolidating the systems, the corporation decided to retire the independent operating company specific intranets into a single intranet for the whole corporation. The corporation also had invested millions of dollars in SOA products that the technical architecture team was keen in finding a project for the initial launch. The intranet consolidation project was such a project that could serve as a launch pad for the SOA products that included the portal software.

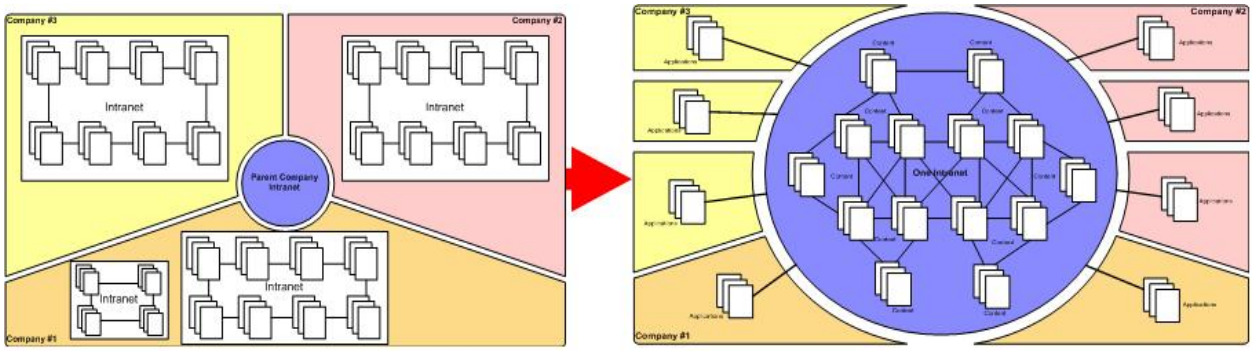

Figure 16 – Consolidation of multiple intranets into a single intranet

Scope: The goal of the project was to consolidate all the operating company specific intranets into a single intranet portal. The executives also agreed to use this project to introduce the existing investment made in the SOA products like the portal server. One of the main reasons to leverage the portal server was the perception from the vendor that a portal would enable business users to directly publish content without any assistance from the IT staff. The business side also made it clear that majority of the content that would be published would not be dynamic in nature; that is it would not be generated from any business applications instead the business will itself author it manually. In other words the content will be static in nature.

Solution: The IT team assigned to this project started working on it with the help of the vendor. Due to the lack of expertise and training internally, expensive vendor professional services were used for the initial launch. As the project progressed multiple issues emerged:

1. The plan was to enable the business users to directly publish content with minimum IT support; therefore it was necessary that quality assurance (testing) environments were in place for the business to use as a sandbox.
    a. This identified the need to purchase more software licenses and hardware and translated into very high additional costs for the project.
2. The content management component that would have enabled the business to publish content was not very user friendly.

a. After business users went through the whole workflow as to how to publish content in the different real estates of the portal environment, it became very evident that extensive training was needed for the business to correctly use the tool.

3. The initial load and performance testing also indicated that as more users were going to start using the new intranet, the current hardware environment would have to be scaled.

    a. Hardware scaling would also require more software licenses (additional cost) since licenses are normally CPU based.

There were other issues as well but the above ones were critical enough that the IT and business both decided not to use the SOA portal product and instead use a normal web server to publish content that was needed for the consolidated intranet.

Conclusion: This case study clearly shows that even though products were already in-house, implementation cost, performance needs and training were the main hindrances in implementing the SOA model in the environment.

Project Case Study # 5: Credit Check Service

Background: This case study deals with a company that has leveraged a point-to-point integration pattern with a credit agency in order to get a credit score before selling their products to the customer. The need to enhance the existing capability came from the fact that the existing interface and technology were no longer supported by the credit agency. Therefore, the existing component could not be leveraged. The company also realized that in future there will be more business applications requiring access to customer credit information.

Scope: A project was launched to provide the customer acquisition application to call the credit agency new application programming interface (API) which completely adheres to the web service standards. Web Service is considered one of the technologies completely in alignment with SOA. The requirement for the project was to ensure that the business logic for the credit service is reusable and is segregated from the technical implementation such that if the company changed its credit agency or the credit agency changed its interface the impact would be minimal.

Solution: There were three options available to implement:

1. The application that needs to leverage the credit agency functionality would implement both the business rules and connectivity to the agency.
2. The connectivity service is independent, but the business logic is still part of the application needing the credit functionality.
3. The business logic and connectivity are completely independent from the application needing the credit functionality; there were two options for this as well:
    a. Connectivity and business logic functionality is one unit of work that is reusable.
    b. Connectivity and business logic are two separate individual units of work that are reusable.

The conceptual solution option that was selected was 3b which was in complete alignment with the SOA model. Business logic and connectivity functionality can be considered as individual services on a conceptual level, but would be completely dependent on the implementation design as to whether they are truly SOA services. Unfortunately, the solution implementation design chose to put the business logic into a mainframe module where the application needing the credit functionality resided. Even though the business logic module

was independent, it was not reusable. The connectivity service was implemented with SOA implementation design that service was truly independent and reusable as it used the web services standard.

Conclusion: Conceptually, the design was correct but due to the lack of governance within the software development lifecycle, the implementation design did not align with the initial conceptual design. Other reasons for this direction were:

- Timeline in which this was needed
- For IT staff it was easier to implement business logic in a mainframe module

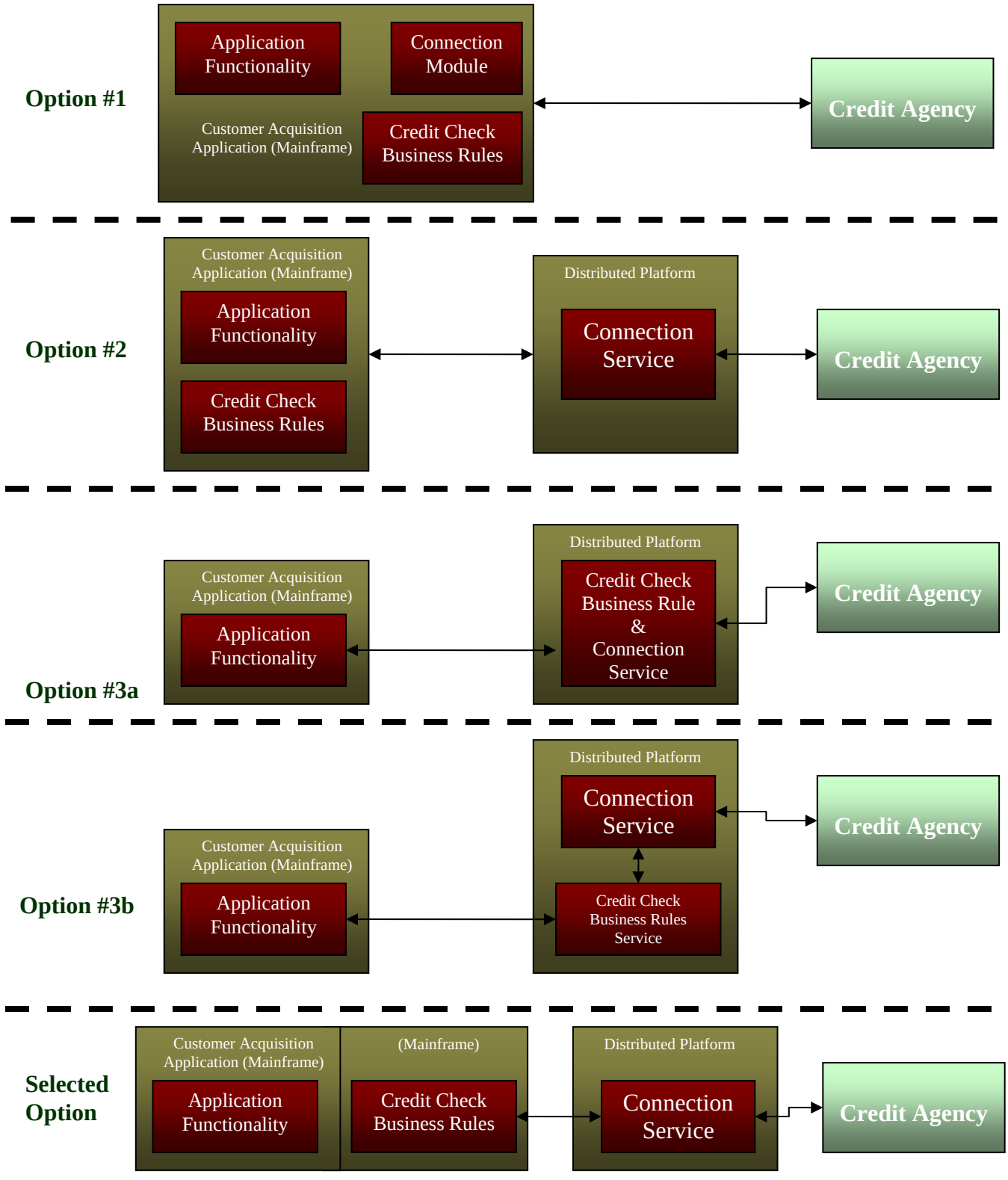

Figure 17 - Different options for implementing Credit Check functionality

Process and Methodology

In order to determine the feasibility of SOA architecture in a legacy environment, one should analyze the history of SOA implementation using both published works as well as studies of actual implementations. Talking directly to project teams and beneficiary organizations will aid in truly understanding and analyzing SOA implementations.

In this paper, analysis of eight case studies are documented. Five case studies are directly from SOA projects where project team members were available to provide information on the scope, requirements and solution of the project. The other three case studies are from industry published sources which are pertinent to SOA. The main reason of using this approach is that case studies not only provide some statistical and experimental value but it directly relates to actual implementation and the lessons learned. Wikipedia [9] provides a good reason for using this methodology, it documents that:

> *"Rather than using samples and following a rigid protocol to examine limited number of variables, case study methods involve an in-depth, longitudinal examination of a single instance or event: a **case**. They provide a systematic way of looking at events, collecting data, analyzing information, and reporting the results. As a result the researcher may gain a sharpened understanding of why the instance happened as it did, and what might become important to look at more extensively in future research. Case studies lend themselves to both generating and testing hypotheses. Another suggestion is that **case study** should be defined as a **research strategy**, an empirical inquiry that investigates a phenomenon within its real-life context. Case study research means single and multiple case studies, can include quantitative evidence, relies on multiple sources of evidence and benefits from the prior development of theoretical propositions. Case studies should not be confused with qualitative research and they can be based on any mix of quantitative and qualitative evidence. Single-subject research provides the statistical framework for making inferences from quantitative case-study data".*

Inference

After evaluating the different case studies mentioned in this paper, it is clear that Service Oriented Architecture (SOA) is much more than an architecture style, a pattern, a strategy, a programming model, a set of technologies or standard based infrastructure; it is truly an IT culture change. What does it mean by "culture change"? If one looks at the different definitions that are given for SOA, it is clear that the underlying goal for SOA is modularization of components such that they become very reusable. This concept is truly not that new. As soon as the subroutines and functions were invented in the computer programming world, the idea for modularizing the capability was invented. The advent of the internet has definitely given this idea a new twist since there are so many technologies being developed that every one is wondering how to integrate and thus SOA provides that answer by standardization of a pattern. The question becomes why the concept of SOA has not been achieved yet, if this idea has been thought and worked upon for decades. The answer is "SOA is an IT culture change". In a normal IT organization, implementation of new technologies is based upon projects and projects are created based on business needs. If one tries to implement SOA as a project, then according to a reputable SOA author [12],

> *"Because SOA is the approach, not the solution, cost justifying it in the context of a quarterly budget is tough – and perhaps even counter-effective".*

This is actually quite evident in the case studies documented in the paper, since the bottom line for a project is to achieve the business objective and if the SOA implementation is more expensive initially than other approaches and IT has a limited budget, then adopting SOA is very hard. This is especially true in an environment where integration is point-to-point and no middleware layer abstracts the application's interfaces that are being integrated as

documented in this paper where EAI benefits are mentioned. Gartner [15] one of the premier IT research said,

> *"SOA is useful in systematic application engineering, especially in circumstances where request/reply is the natural relationship pattern between software components. However, SOA comes at the cost of additional design and development efforts, and it also carries certain middleware coupling requirements. Limited-scope applications, intended for short life, typically do not benefit from SOA because their business logic is neither reused nor changed during their useful lifetime. SOA is also not recommended for applications using asynchronous one-way interactions. Here, the loose coupling of SOA is unnecessary and undesirable."*

This shows that SOA is not a silver bullet that can solve all the problems that are faced by businesses today. Aberdeen Group [13] which is also a reputable corporation in the world of IT research said that "*SOA is hard to do*", they said this because the lead research analyst documented that

> *"As the lead for Aberdeen's fact-based research on SOA, the analyst looks at data he has gathered from a thousand-plus companies and concludes that if 2005 was the year of SOA romance, by the end of 2006, the honeymoon is over. The promise of SOA cost-savings remains its biggest lure, he said, looking over the research he has gathered during six trips out into the field where he visited IT professionals who are actually trying to make SOA work. The cost of integrating applications is roughly 40 percent of the IT budget," Kastner said. "That's 40 percent of a $1.3 trillion global IT budget. It's just an enormous factor in overall costs".*

The result of this research from Aberdeen Group and the data from the case studies clearly shows that SOA initial cost is very high and therefore a normal IT project that is trying to get SOA implemented will have a very hard time justifying it.

Does this mean SOA is not the right direction? The answer is definitely “NO”, it is the correct direction but the current IT philosophy of how business creates projects for IT and how software development process supports the business does not fit this model, therefore “SOA is a culture change”. In many IT organizations, the waterfall model is adopted as the way of doing software development. SOA in an environment which is based on waterfall is very hard to implement. A more iterative approach is needed which is normally part of extreme programming or agile development methodology. This alone cannot solve the initial SOA implementation issues. The project initiation process for IT also needs modification so that in addition to normal project work, there is also an agreement between business and IT that non-project development will occur as capital investment into the IT SOA infrastructure piece by piece. This means that business and IT organizations have to become visionaries so that they can predict the capabilities in advance such that services that support those capabilities are created as an ongoing non-project IT development.

Now the question arises, how can this work? This can easily work if the principles of enterprise architecture methodology are followed. Currently in most IT organizations there is a department for Enterprise Architecture which is normally responsible for defining the strategy for the business applications and IT technologies portfolio of the company. This department is normally part of the IT organization. In order to make it more influential for SOA implementation, this department needs to be a matrix organization reporting to both IT and business. The main reason is that most of IT departments are run as a business unit and are completely dependent on the business side for funding. Therefore a culture change is

needed where the vision of the enterprise is understood by the business from an IT strategic viewpoint and this can only be achieved through a group which is responsible for defining the strategy for the enterprise. Enterprise Architecture under IT has made sense to many, since one of the goals of defining strategy is to understand the business first and create the roadmap as to how to implement it for the company. IT is definitely part of the roadmap but it is not the driving organization; therefore keeping this strategic group only within IT does not make sense. With that said, SOA implementation with this model becomes are little easier since the Enterprise Architect can now create a system plan which is based on the strategic vision and can allot dollars to service creation even if they are not project specific. The Enterprise Architect will be able to do this because they are now also part of the business organization and will have more influence in how business dollars are spent. This could eventually assist in increasing IT productivity as the Enterprise Architect also understands the nature of how IT operates.

This notion is in complete alignment with the main reasons of why SOA fails as documented by David Linthicum of InfoWorld [14]. He states that the main five reasons for failure are:

1. *The enterprise considers SOA a project versus what it is; a more holistic notion. Thus, management thinks they can implement an ESB or other SOA technology, and they're done. I've been ranting about this enough that I won't do it here. However, just to recap, you need to determine your requirements first, and then the technology that works to solve the problem. Common sense, one would think, but more enterprises are failing this way, or will fail this way.*
2. *They use 2nd tier talent. They attempt to implement SOA using people who really don't understand the concept of SOA, and perhaps never will. We know who these*

people are. They must be stopped. It takes education to get this stuff, and those who lack the education yet are making critical decisions are down right dangerous.

3. *They are under-resourced. "Go make huge sweeping changes within our IT infrastructure, and do so with about 10 percent of the resources you really need." You can't do this on a budget, unfortunately. You have to lubricate SOA projects with money.*
4. *They allow the vendors to define their solution. The vendors don't understand your core business issues, and really have a conflict of interest. When you sell a hammer, everything looks like a nail. Work with vendors, make them understand your requirements, but you're ultimately responsible for the solution.*
5. *Requirements are not fully gathered. There is no domain understanding before the solution is developed. I've been ranting about this as well; I'll leave it alone. Use them, please.*

SOA Implementation Guideline

Based on all of the information discussed to this point, it is imperative that guidelines are crafted for IT organizations that are still leveraging a legacy environment so that they have a tool that can assist them in making decisions as to when and how to enable SOA in legacy environments. This section basically focuses on the key areas that have to be addressed before a company embarks on SOA journey.

Key Success Factors

The key success factors for SOA are:

- Leadership: Executive commitment to the "culture change" ideology and appointment of a key executive as the Lead SOA Evangelist.
- Enterprise Architecture: This represents not the actual department or team but the role it plays in defining the strategy and roadmap for the successful implementation of SOA and includes the establishment of architecture guidelines and enforcement of standards.
- Funding: As obvious from the case studies, initial investment is the main hurdle for legacy environment to move toward SOA. Therefore, figuring out how to create a convincing business case is critical.
- Software Development Lifecycle (SDLC): Major process re-engineering is needed to mold the legacy oriented SDLC to the new SOA model.
- Technology: This relates not only to new SOA specific technologies but also how legacy technologies can be re-architected to enable SOA principles.

- Training: This not only related to getting acquainted with the new technologies but also a mindset change of the workforce.
- Governance Model: Without this there are no check and balances to ensure the correct implementation from initiation to execution.
- Portfolio and Pipeline management: Execution of the strategy and the roadmap created by enterprise architecture for efficient sequencing of the projects.

Enterprise Architecture

Enterprise Architecture organization is responsible for understanding the business goals and strategies along with the IT capabilities available to the enterprise. This information is necessary for defining the roadmap as to how to move from the current architecture to a target architecture which aligns with the business strategy. The roadmap then assists in planning the projects that are needed for the fiscal year as well as defining the long term plan. This plan leads into many other areas where enterprise architect provides guidance to ensure the approach and execution is in alignment with the roadmap and the business strategy. IBM's diagram [16] in Figure 18 shows on a high level the different facets of enterprise architecture. Enterprise Architecture is therefore the only team which has the necessary visibility to both business and IT objective and understands how to best align them. This serves the SOA cause the best, since if enterprise architects become the leaders in providing the guidance needed to implement SOA, then their role itself provides the tools necessary to convince not only IT but business as well. Enterprise Architecture is therefore the enabler for SOA in any enterprise. With that said, Enterprise Architecture also needs to evolve to accommodate the needs of SOA. This would require better alignment with the business

organization. This could be achieved by either moving the enterprise architecture team completely under the business organization or at a minimum making it a matrix organization to business. Along with this, enterprise architecture team should lead the effort of establishing SOA center of excellence within the enterprise. This concept is discussed in detail in later sections.

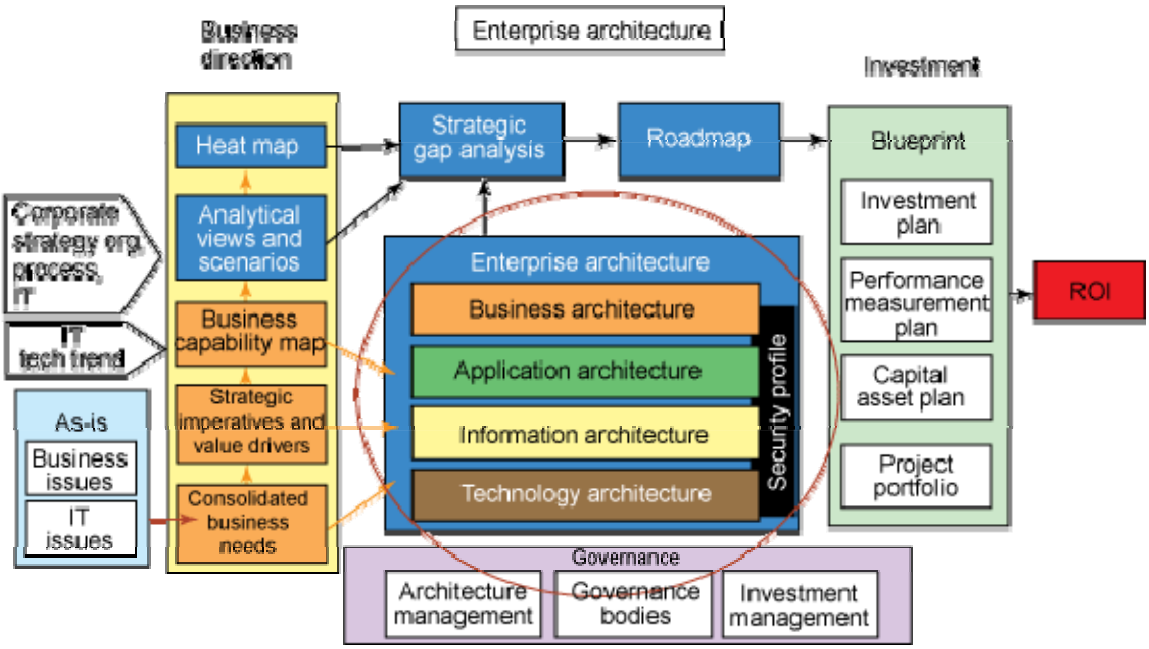

Figure 18 – IBM pictorial definition of Enterprise Architecture

Aberdeen Group survey (Figure 19) about who is leading the SOA effort clearly shows the most are opting to architecture to lead the effort.

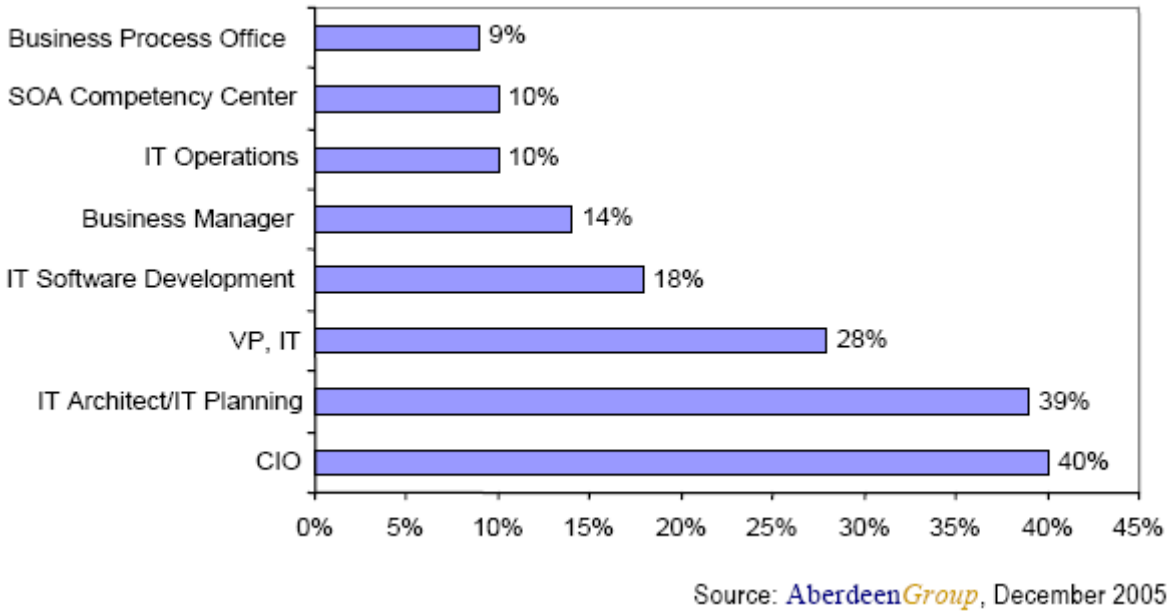

Figure 19 – Aberdeen Survey: Who is leading SOA?

IT project lifecycle

Based on the case studies and the other relevant data documented in this paper, the following is a sample life cycle recommendation that separates service creation from normal project initiation. This approach is fundamental change for legacy environments where budgets are tight and tactical solutions are prevalent to meet business strategies. This model makes the business side more proactive in understanding impacts of their strategy to the overall business environment which includes IT implementation environment.

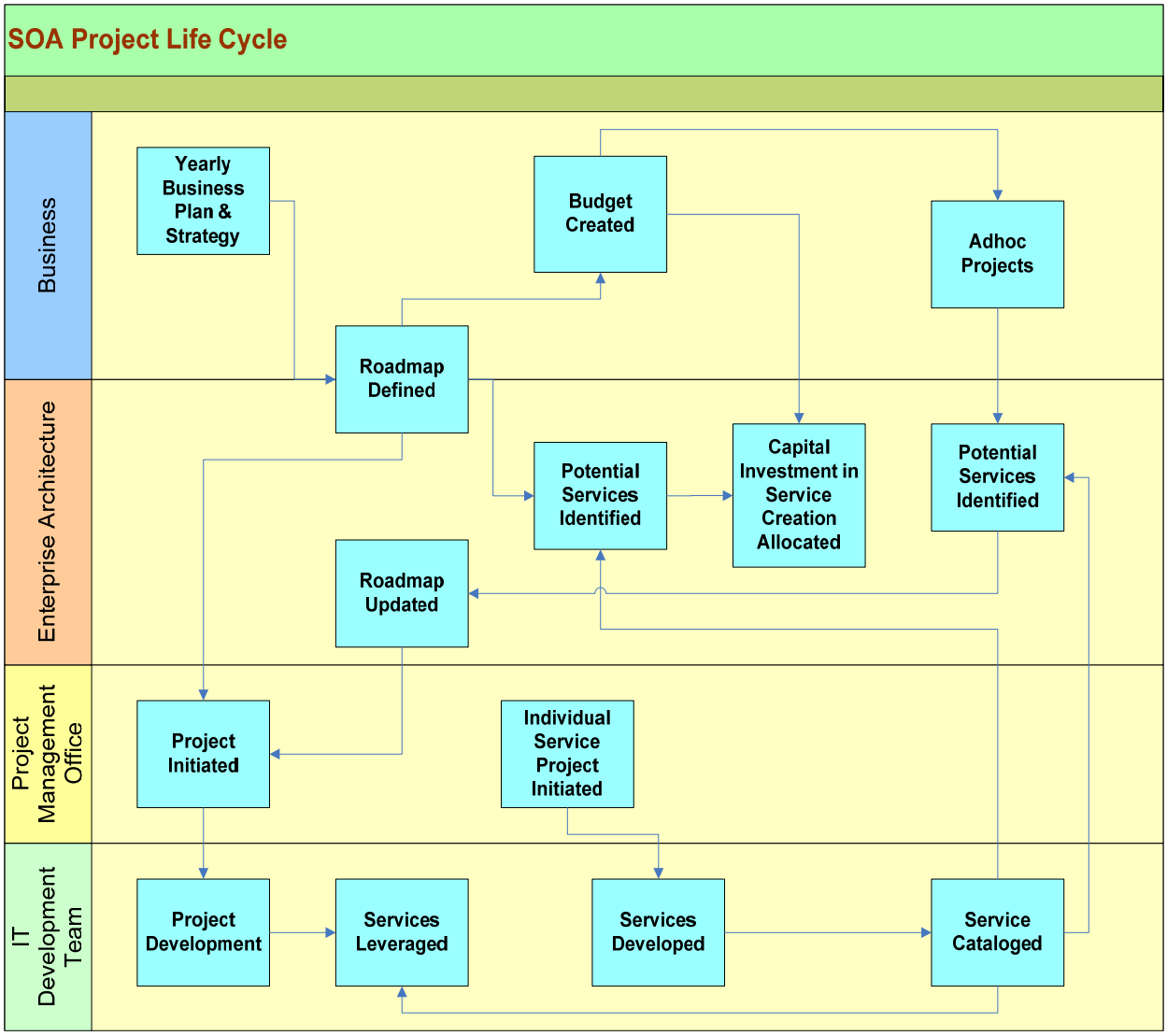

Figure 20 – Sample SOA Project Life cycle

SOA Governance

Governance is the concept where certain individuals are assigned the responsibility of enforcing rules and policies for achieving a particular goal. It is the decision making and accountability framework for the organization. In the case of SOA, governance is very critical since services and their associated specifications are created to define how to leverage them. Without a governing body that can enforce SOA creation and execution guidelines, the

benefits of SOA cannot be achieved. Therefore, there has to be a framework defined for decision making, and responsibility has to be allocated to individuals who should be making decisions. The process around how this decision making framework will work and how to effectively monitor its progress is also needed.

Governance can be centralized or decentralized. For legacy environments the recommendation would be to have a centralized group created to perform that duty. The concept of center of excellence is therefore important in the SOA world and it is discuss in detail in the next section.

Center of Excellence

Center of Excellence is an organization created to define and maintain the standards for implementing a particular technology, process or any other activity. This organization can be virtual where experts from other organizations are chosen to form a virtual center of excellence. In the case of SOA center of excellence in a legacy environment, people should be chosen who are most acquainted with SOA principles and have experience in implementing SOA related processes and technologies. The role of the center of excellence would be to:

- Setup standards and guidelines for SOA for the enterprise
- Enforce standards
- Manage the service life cycle
- SOA Technology strategy definition
- Manage the SOA infrastructure
- Manage SOA training

In a legacy environment, enterprise architecture team can initially take over the center of excellence responsibility but in time should add other members as SOA matures in the

organization. The governance will start on a smaller scale in the beginning but efforts should be put in to grow its influence so that as services are created their use is correctly governed. An example is shown in Figure 21 from a legacy environment perspective where the Enterprise Architecture team took over the initial implementation of the governance and put an initial model in place to start the governance process.

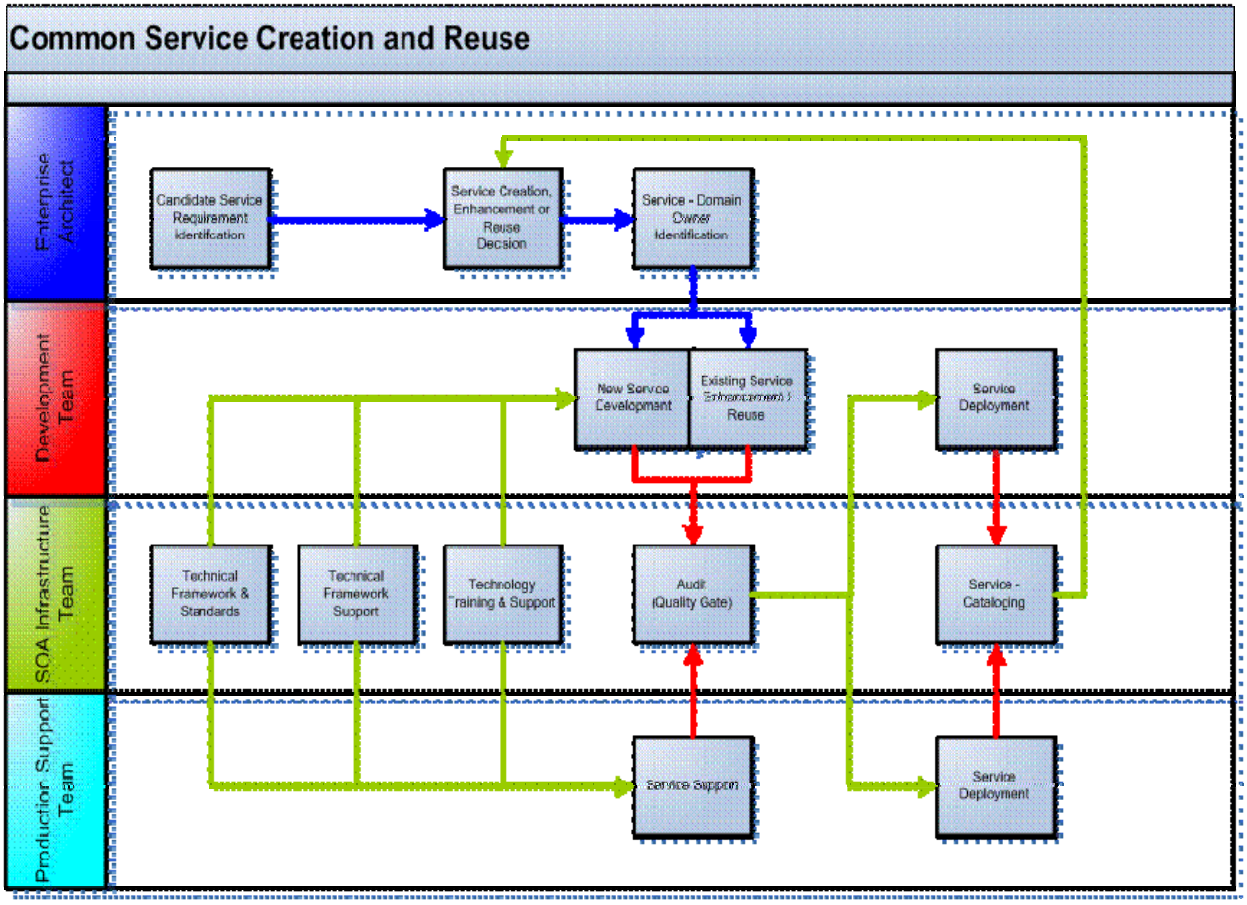

Figure 21 – Initial Governance model implemented at a legacy environment

This governance model is not comprehensive since is does not define specific software development life cycle quality gates to enforce and validate the overall process of implementing services instead it has minimum quality gates for only audit purpose. It relies on the enterprise architect and the SOA infrastructure team who are part of the center of excellence to enforce the SOA guidelines and polices. The concept of a service librarian is

implemented in this model where the infrastructure team is cataloging the services and the enterprise architect is responsible for it reuse applications. This whole process can only be successful if the project pipeline management team ensures under the software development life cycle that every project is evaluated by the enterprise architect to ensure correct service reuse. If the enterprise architect does not get that visibility, then reuse will not happen and the cost of implementing SOA will increase.

There is also another aspect that is invaluable in assessing the benefit of implementing SOA. This aspect is the center of excellence responsibility of defining and measuring key performance indicators (KPI) for service creation, reuse and execution. An organization will never truly know how well its implementation of any technology or process is going until they measure and compare it with the expectation that should be defined as yearly objective for not only the center of excellence but also other organization that are part of the SOA implementation.

SOA Legacy Environment Pattern

The key characteristics for a SOA pattern in a legacy environment are:

- Do not rewrite existing monolithic systems with SOA
- SOA governance is just a subset of the existing IT governance model
- Centralize service creation in a single development team
- Empower enterprise architecture to identify services
- Implement a SOA service catalog but not necessarily a registry technology, this could be manual function through the enterprise architecture team
- Only create service when there are multiple possible consumer of the service

- First evaluate existing technology before deciding on buying new SOA specific technologies
    - The goal is to have an abstraction service layer that is reusable; this can be achieved with pre-SOA technologies also, like EAI.
- Detach service creation from normal project life cycle

Figure 22 captures the pattern that legacy environment should adopt in order to at least leverage some of the benefits that the SOA industry is promising.

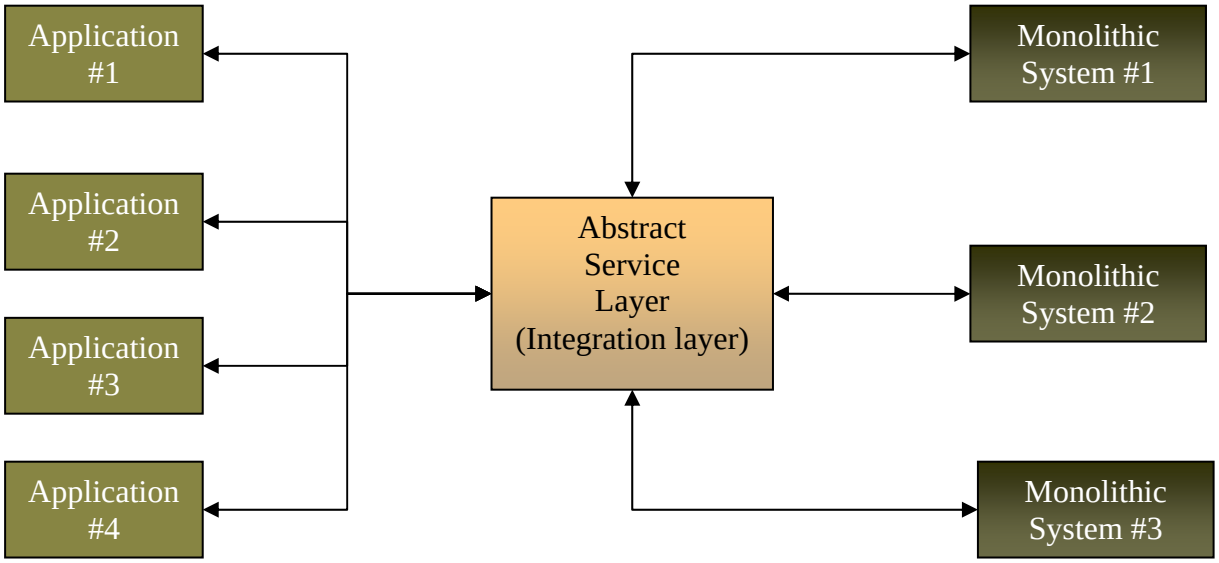

Figure 22 – Legacy environment SOA Pattern

Conclusion

IT organizations that have a legacy environment should be very careful in implementing SOA. As mentioned before, enterprise architecture is key in SOA implementation. Therefore, experienced IT professional staff that understands the charter of enterprise architecture should be involved in the initial process and guidelines documented in this paper should be followed to evaluate the feasibility of the SOA implementation. There is definitely a misperception that SOA is the best technology out there that can solve everything. In certain integration issues, it definitely has its benefits but it also requires a large investment of both capital and process re-engineering. Legacy environment should try to segregate the implementation of SOA within their enterprise so that it exposure initially is with very limited number of teams. After the initial benefits are visible, then SOA should be evaluated for enterprise wide implementation.

SOA can be implemented without tools that industry vendors are advertising. As mentioned, it is a “cultural change” and not a technology change. Most of the existing legacy environments have the necessary tools today in-house that can assist in moving in that direction. The actual initial cost is training and the impact of process change to existing stable IT operating environments.

Suggestions for Additional Work

This paper is focused on the legacy environment challenges for implementing SOA which includes the initial investment. There has been extensive work both from business process and technology perspective that is now available for free for companies through the open source community. Therefore it will be very interesting to see what impact the open source community can have on SOA implementation in the legacy environment.

There have been also a lot of advancements made in the network architecture arena; it will be interesting to see whether with these advancements is there any value of rewriting the existing monolithic system to SOA based systems.

Bibliography

[1] Buecker, Axel, Paul Ashley, Martin Borrett, Ming Lu, and Neil Readshaw. Understanding SOA Security Design and Implementation, 2nd ed. Poughkeepsie, IBM 2007.

[2] The ABCs of SOA. Oracle | BEA website, http://www.bea.com/framework.jsp?CNT=soa_abc.htm&FP=/content/solutions/soa/basics/, accessed on October 27, 2008.

[3] SOA and BPM FAQ. Microsoft website, http://www.microsoft.com/soa/about/faq.aspx#soafaq, accessed on October 27, 2008.

[4] Service-Oriented Architecture. Wikipedia website, http://en.wikipedia.org/wiki/Service_Oriented_Architecture, accessed on October 27, 2008.

[5] Hohpe, Gregor, and Bobby Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Boston: Pearson Education 2004.

[6] Case studies in implementing service oriented architecture (SOA). ComputerWeekly website, http://www.computerweekly.com/Articles/2007/04/17/222985/case-studies-in-implementing-service-oriented-architecture-soa.htm, accessed on October 27, 2008.

[7] Erlanger, Leon. Verizon goes back to the workbench – Convincing developers to embrace SOA meant extensive custom development. InfoWorld 2005, http://www.infoworld.com/article/05/11/07/45FEsoacaseverizon_1.html?s=feature, accessed on October 27, 2008.

[8] Stuart Sim (Chief Architect – Global Education & research, Sun Client Solutions). Service Oriented Architecture, 11th Semiannual JA-SIG Conference Presentations, Austin Texas, 2005, available at http://web.princeton.edu/sites/isapps/jasig/2005WinterAustin/presentations/SOASims.ppt#266,11,The "Move" to Service Orientation, accessed on October 27, 2008.

[9] Case Study. Wikipedia website, http://en.wikipedia.org/wiki/Case_study, accessed on October 27. 2008.

[10] Keen, Martin, Greg Ackerman, Islam Azaz, Manfred Haas, Richard Johnson, JeeWook Kim and Paul Robertson. Patterns: SOA Foundation – business Process Management Scenario. International Technical Support Organization, IBM, 2006.

[11] McGovern, James, Sameer Tyagi, Michael Stevens, Sunil Mathew. Java Web Services Architecture. Elsevier Science, 2003.

[12] Maguire, James. Does SOA Have an ROI? Datamation article, http://itmanagement.earthweb.com/article.php/3665266, accessed on October 27, 2008.

[13] Seeley, Rich. SOA is hard to do, Aberdeen finds. SearchSOA article, http://searchsoa.techtarget.com/news/article/0,289142,sid26_gci1235685,00.html, accessed on October 27, 2008.

[14] Linthicum, David. Top 5 reasons SOAs Fail. InfoWorld article, http://weblog.infoworld.com/realworldsoa/archives/2007/08/top_5_reasons_s.html, accessed on October 27, 2008.

[15] Natis, Yefim V. and Roy W. Schulte. Introduction to Service-Oriented Architecture, Gartner, 14 April, 2003.

[16] Ibrahim, Mamdouh and Gil Long. A framework for understanding how SOA and Enterprise Architecture work together. http://www.ibm.com/developerworks/webservices/library/ws-soa-enterprise1/, accessed on October 27, 2008.

[17] Freefrom Dynamics Report. Service Oriented Architecture Insights from the front lines. July 2006 Report, http://www.freeformdynamics.com/pdf/SOA-Insights-Jul-06.pdf, accessed on October 27, 2008.

[18] Aberdeen Group Report. The SOA in IT Benchmark Report. http://www.p2080.co.il/go/p2080h/files/8563642203.pdf, accessed on October 27, 2008.

[19] Aberdeen Group Report. SOA middleware starts where web services leaves off. http://eudownload.bea.com/es/events/bpm/soa-middleware-062907.pdf, July 2007 report, accessed on October 27, 2008.

[20] Seeley, Rich. SOA project gets mother of all stress tests. http://searchsoa.techtarget.com/news/article/0,289142,sid26_gci1196264,00.html#, accessed on October 27, 2008.